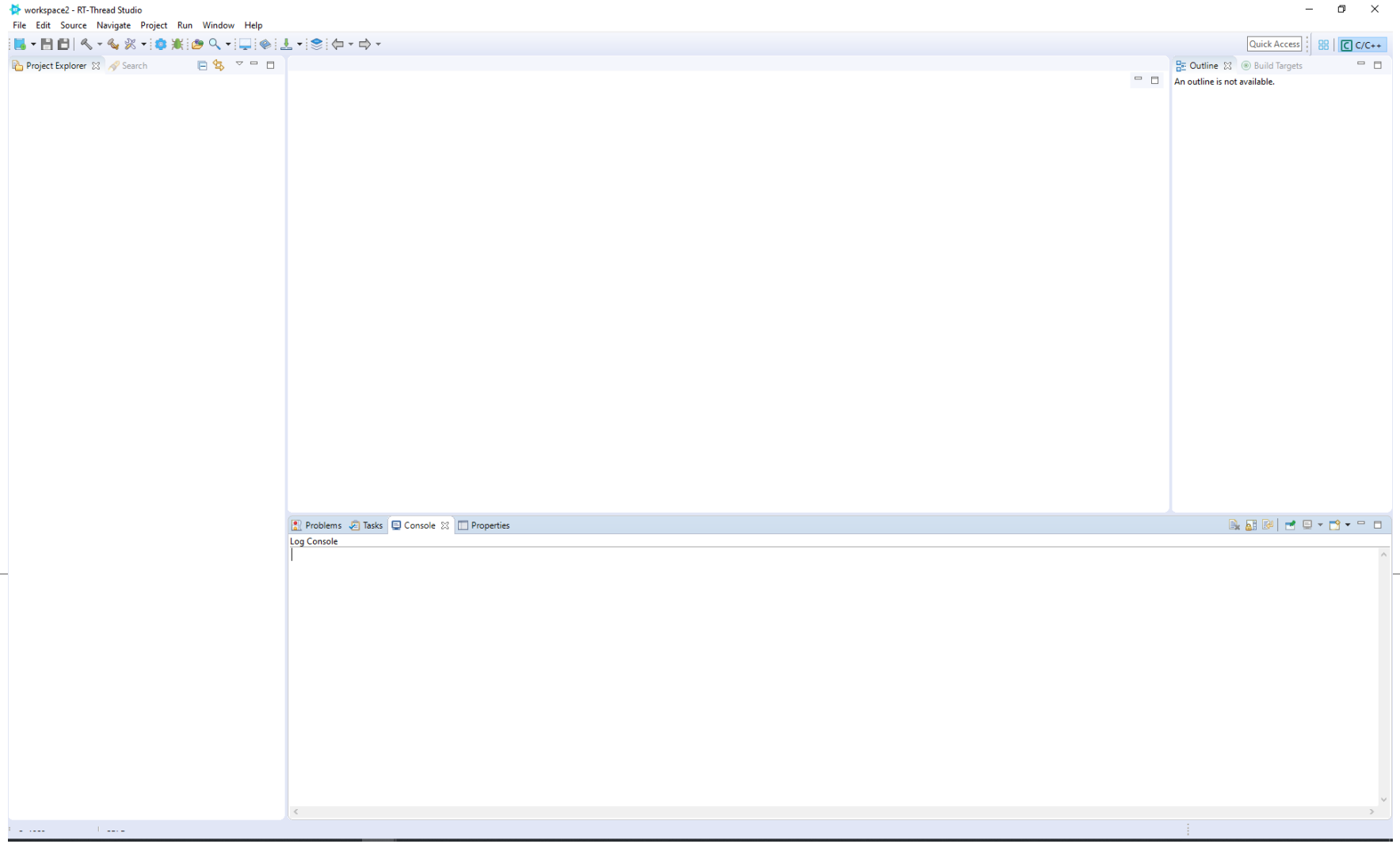
Prosthesis control system using STM32

Development Guide

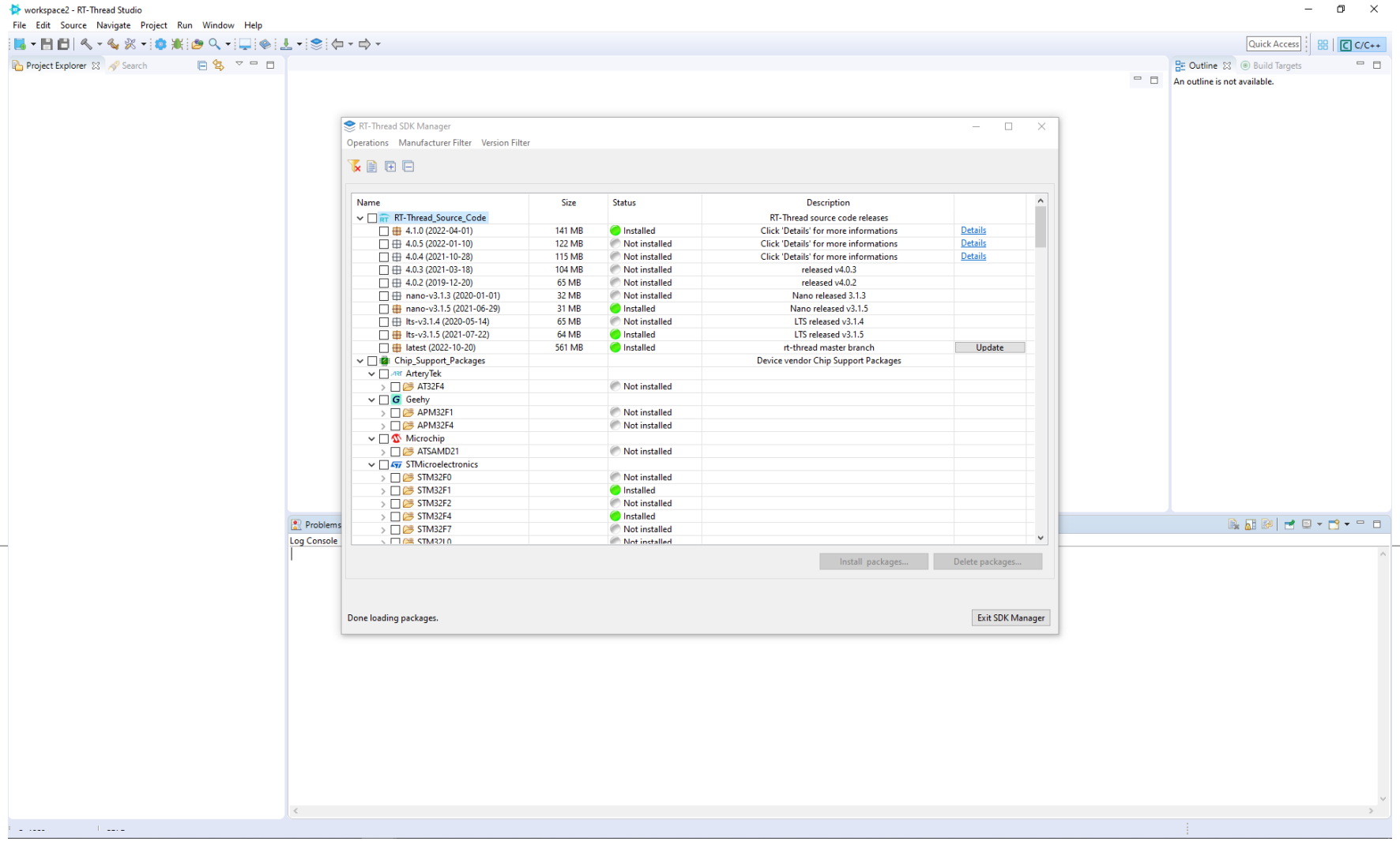
Authors:

- Ph.D. Victor Asanza
- Daniel Montoya

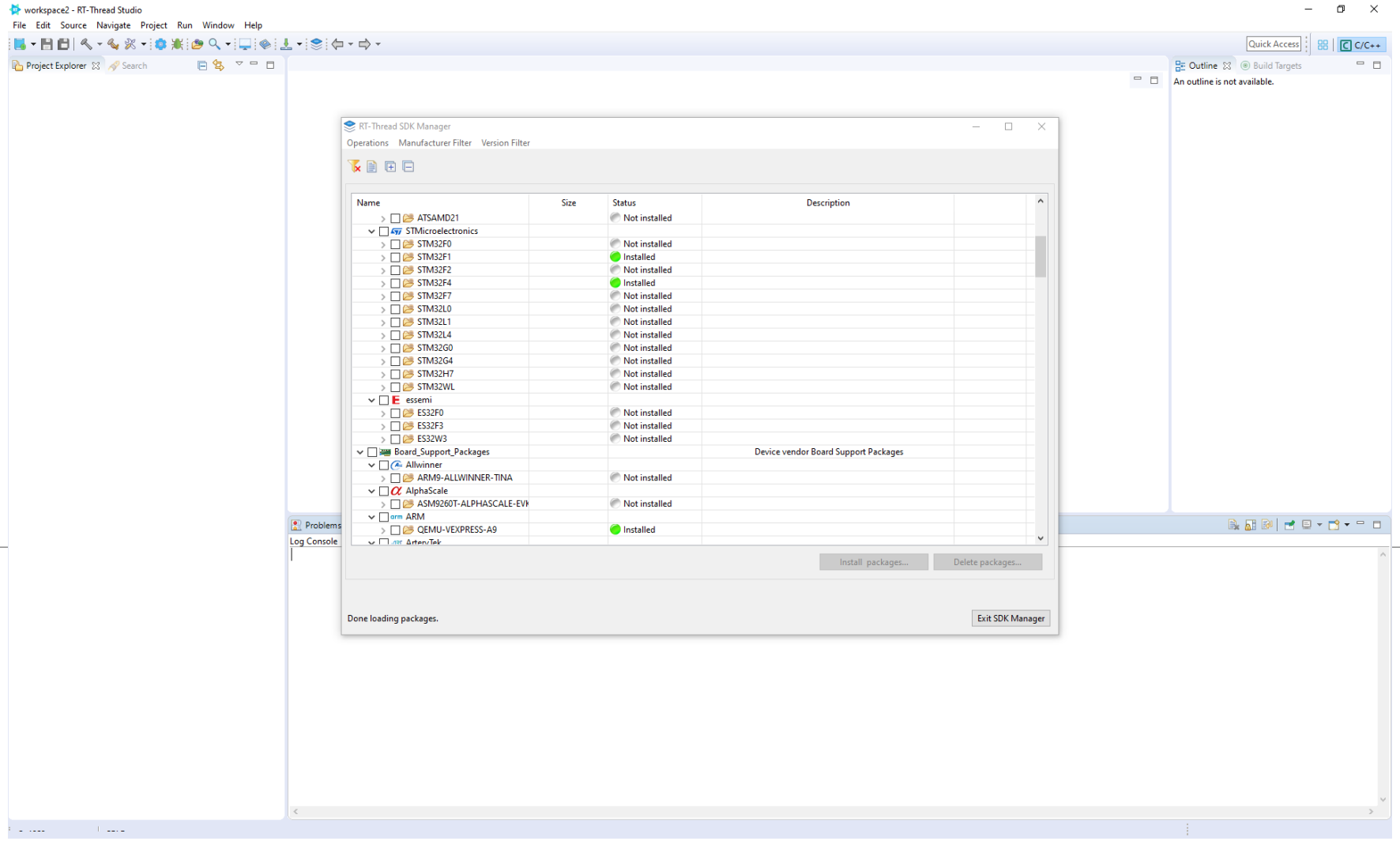
1. Start RT-Thread Studio



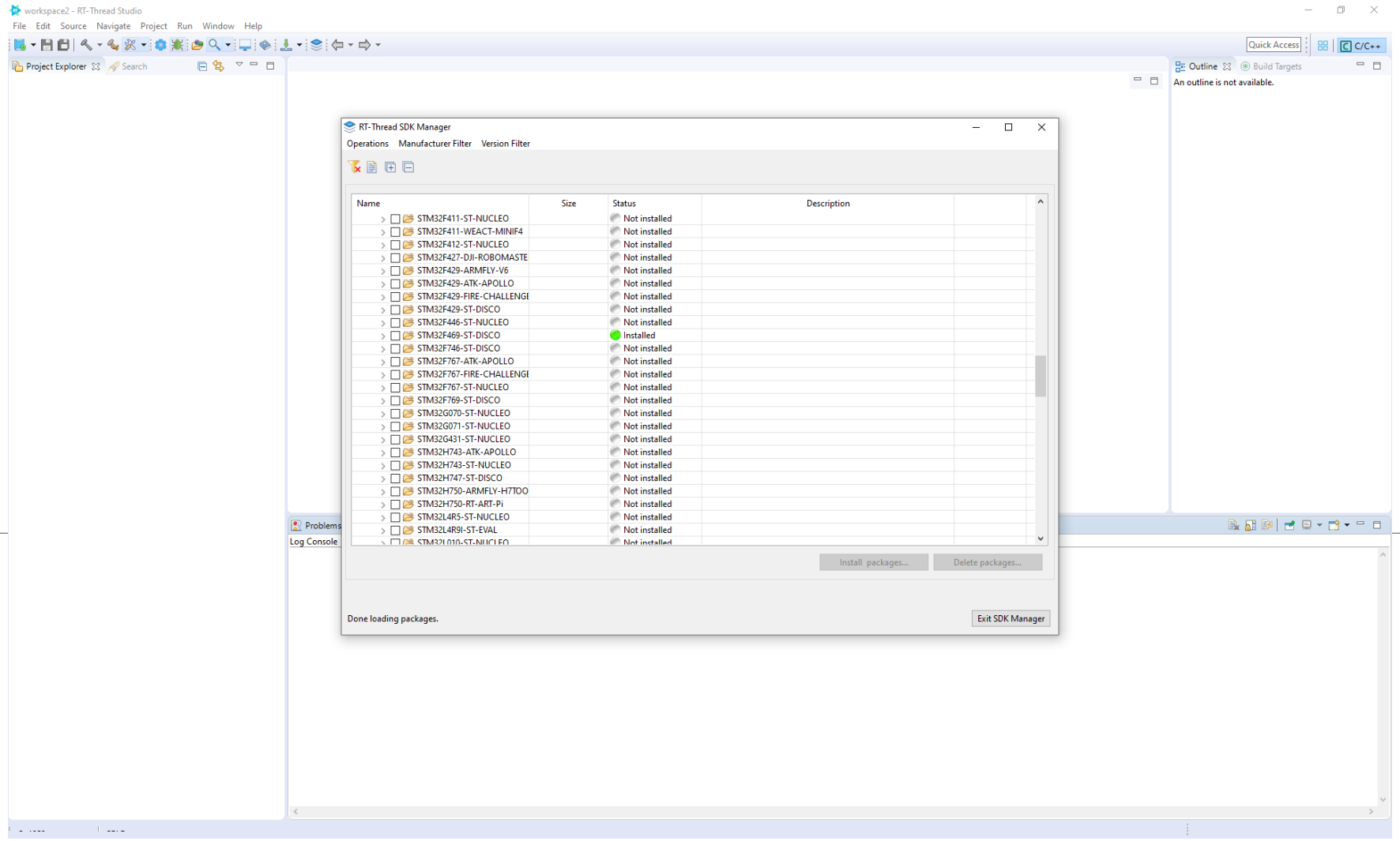
2. Check if the RT-Thread packages are updated



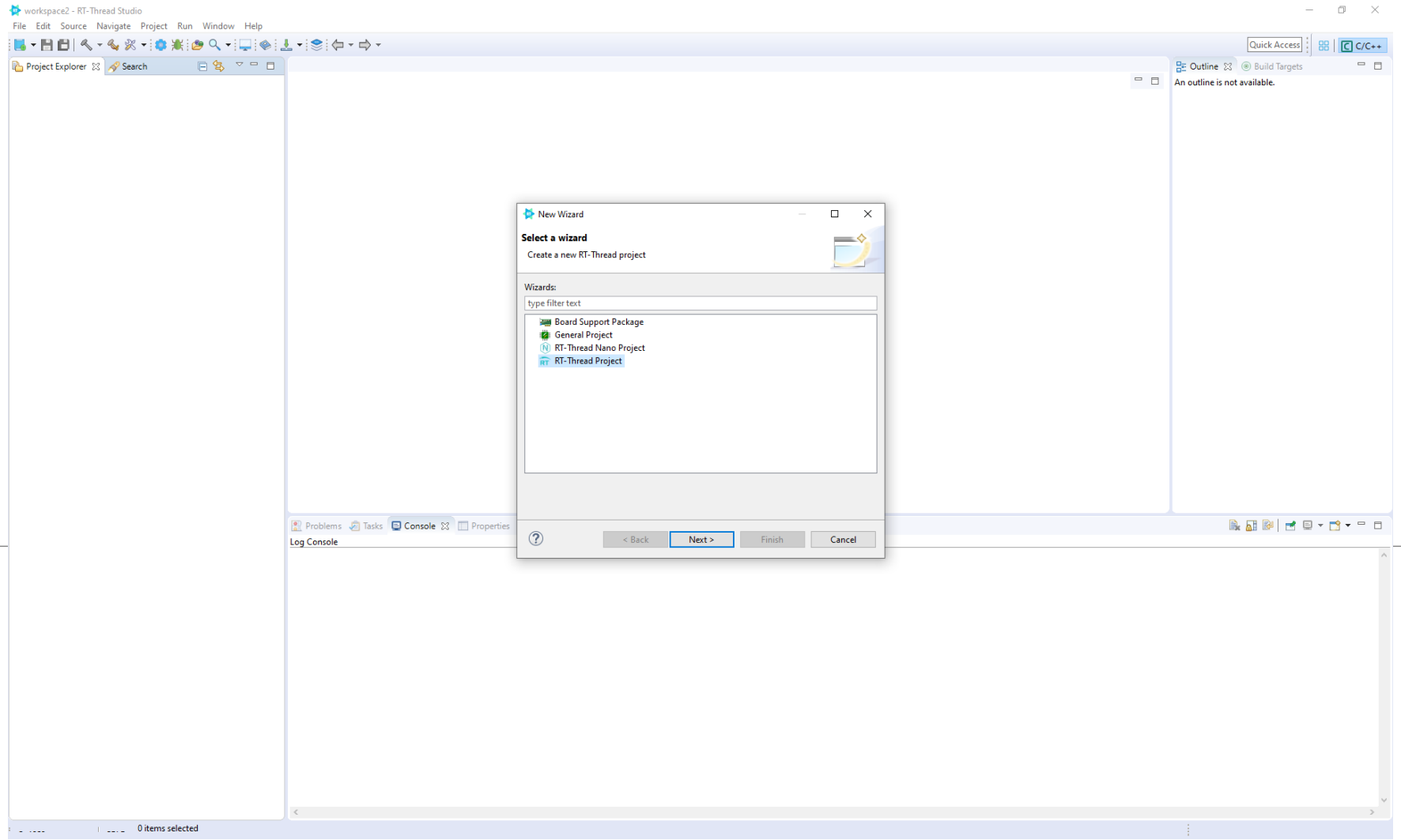
3. Verify or install the packages for the microcontroller to use



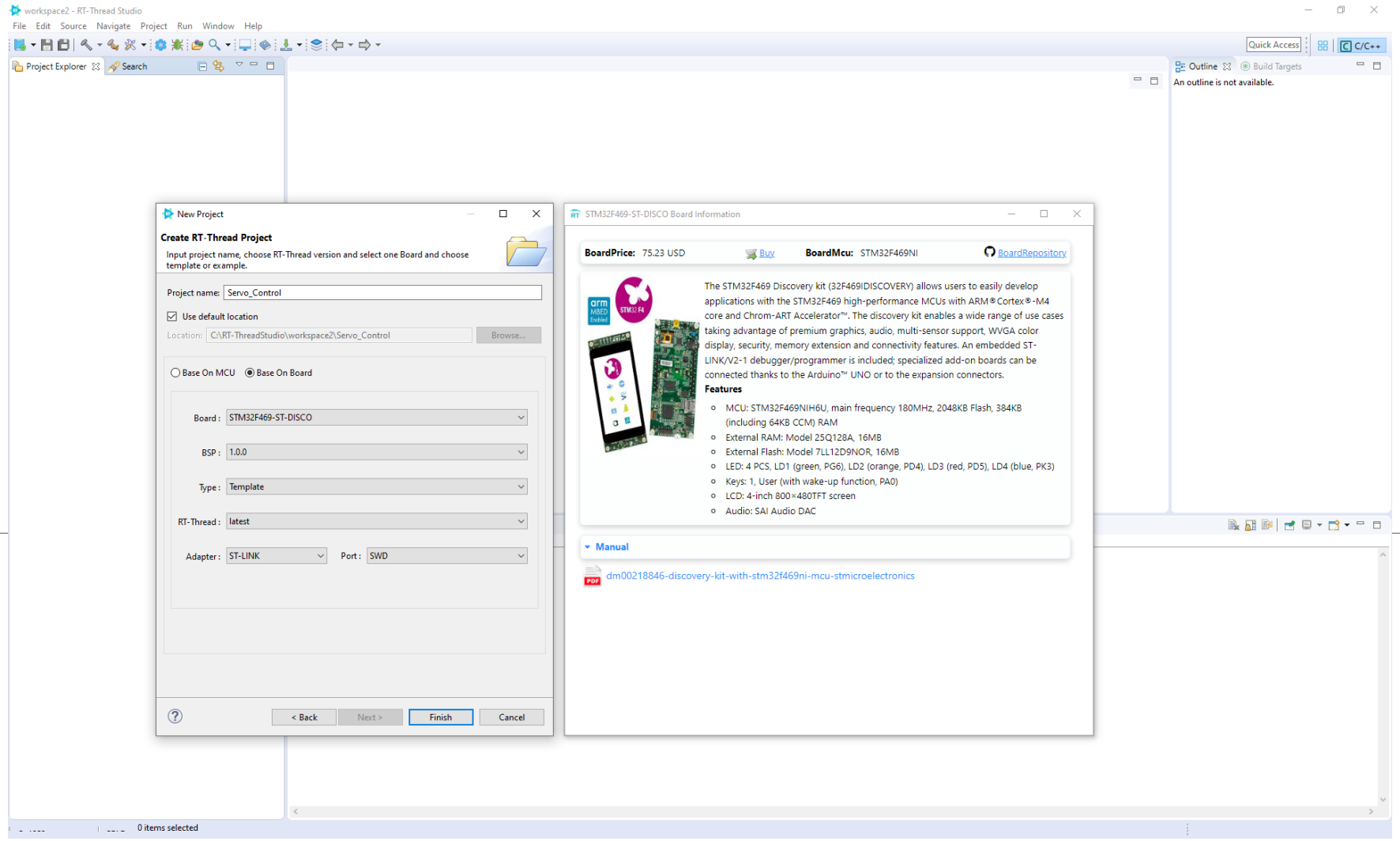
4. Verify or install the packages for the development board.



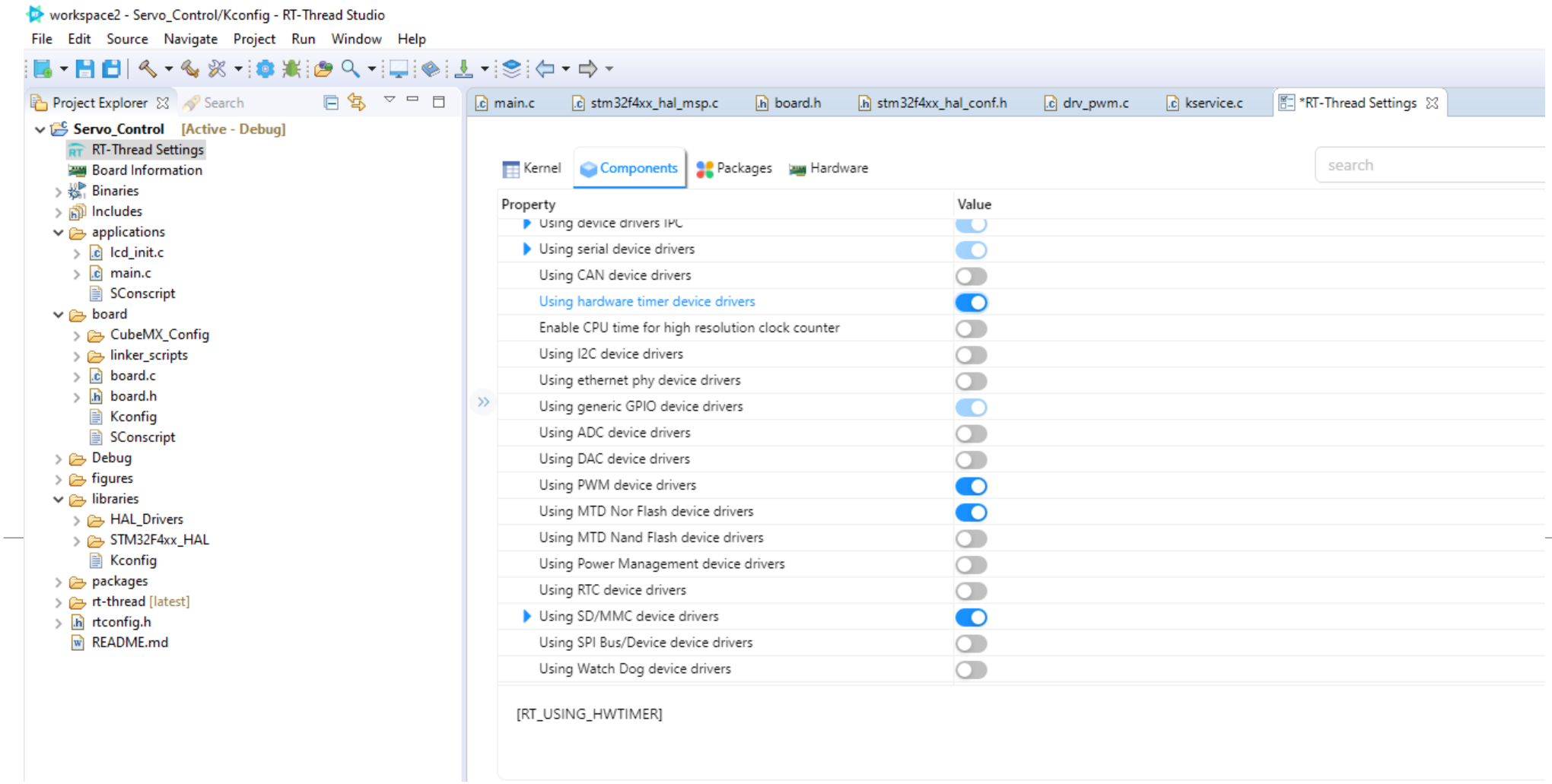
5. Create an RT-Thread project.



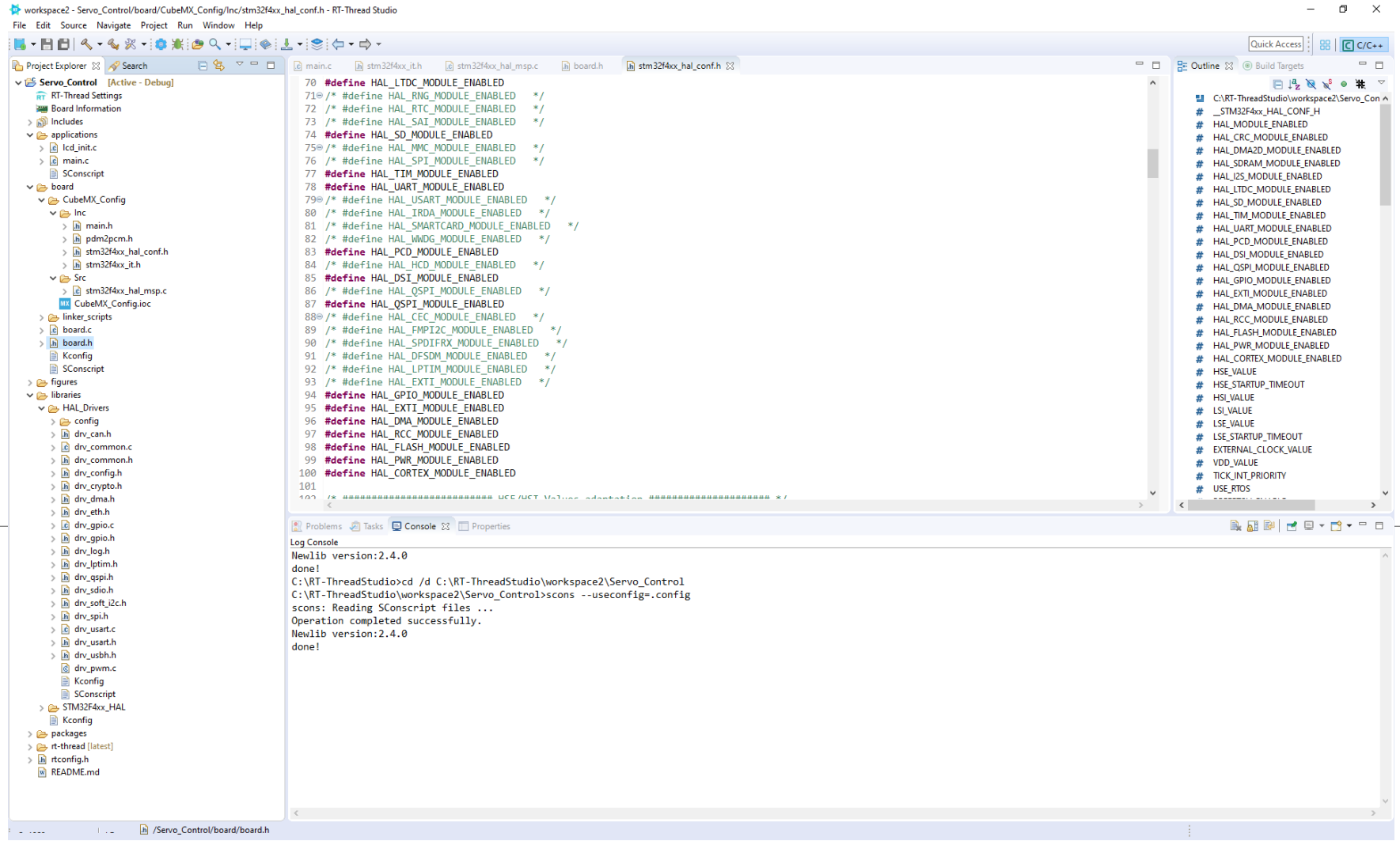
6. Assign a name to the project and choose the development board to use.



7. Enable PWM device drivers and device timers

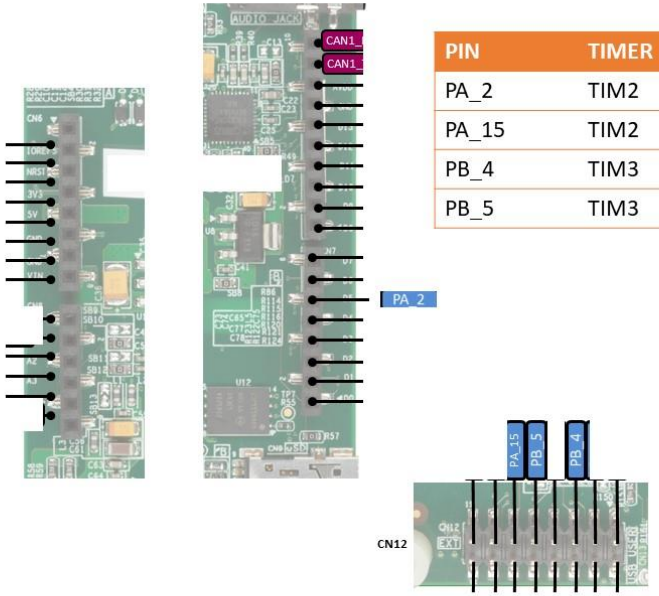


8. Check if the hardware timers and PWM have been enabled on the development board, so we go to the b

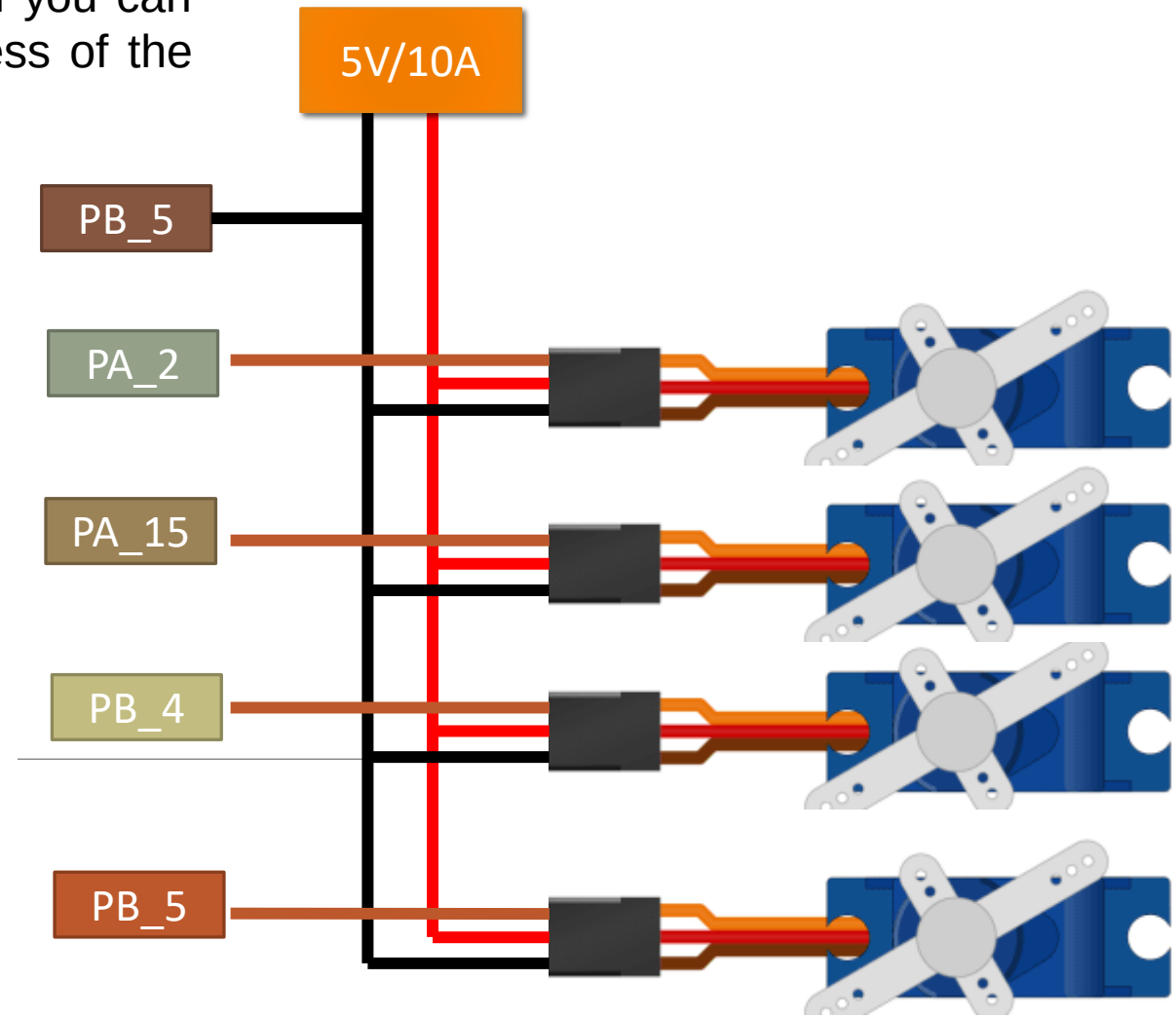


9. Check the documentation of the pins with which you can make the PWM output. Therefore obtain the address of the timers and the corresponding channel..

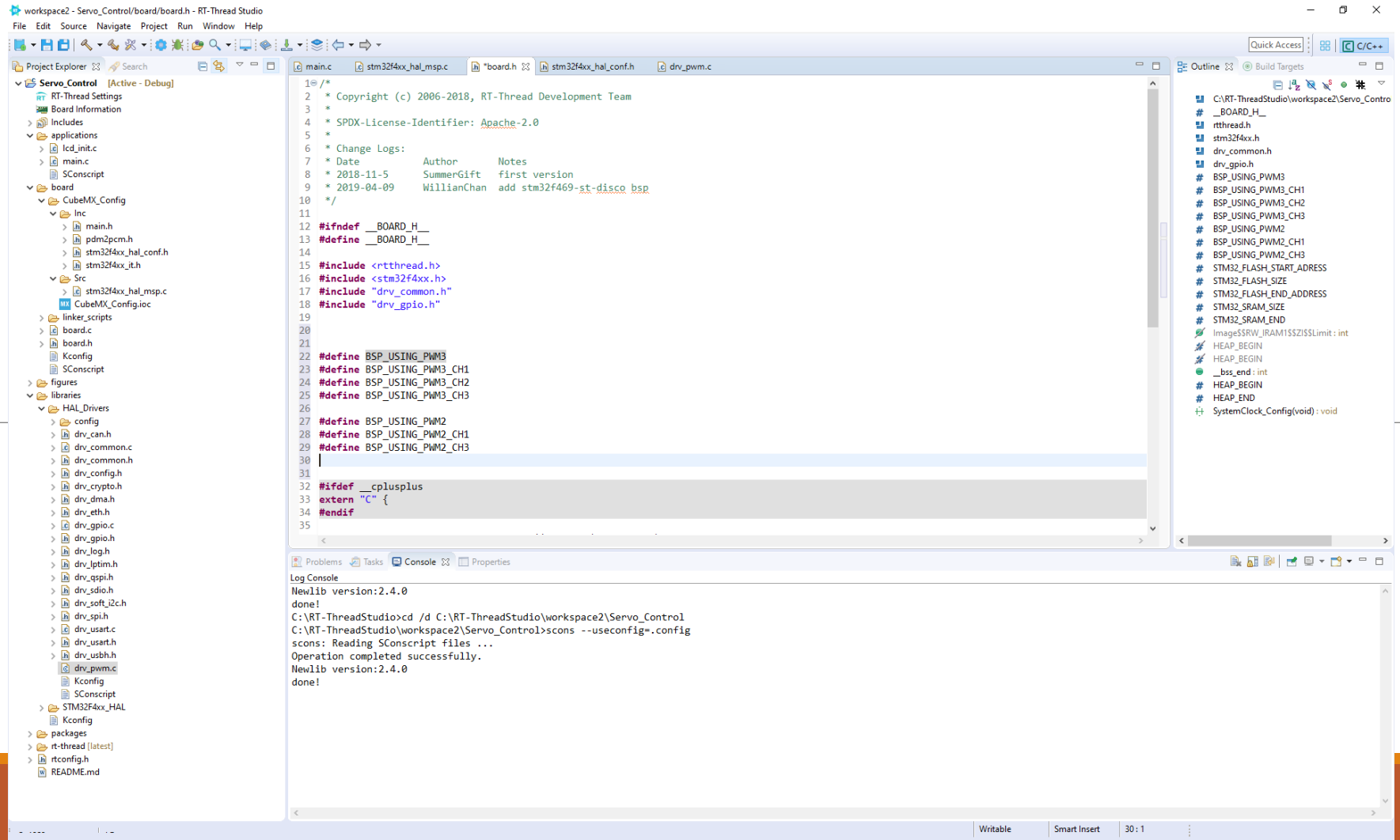
DISCO-F469NI



PIN	TIMER	CHANNEL	FINGER
PA_2	TIM2	CH3	Middle
PA_15	TIM2	CH1	Thumb
PB_4	TIM3	CH1	Index
PB_5	TIM3	CH2	Ring / Little



10. To activate the timers and the respective channels that are going to be used, we go to the board folder, and enter the board.h header file. In which we will include the definitions of PWM outputs 2 and 3, with their respective channels.



11. We check the PWM driver file and we can see that the specified PWM outputs have been activated.

workspace2 - Servo_Control/libraries/HAL_Drivers/drv_pwm.c - RT-Thread Studio

File Edit Source Navigate Project Run Window Help

Project Explorer

- Servo_Control [Active - Debug]
  - RT-Thread Settings
  - Board Information
  - Binaries
  - Includes
  - applications
    - lcd_init.c
    - main.c
    - SConscript
  - board
    - CubeMX_Config
    - linker_scripts
    - board.c
    - board.h
    - Kconfig
    - SConscript
  - Debug
  - figures
  - libraries
    - HAL_Drivers
      - config
      - drv_can.h
      - drv_common.c
      - drv_common.h
      - drv_config.h
      - drv_crypto.h
      - drv_dma.h
      - drv_eth.h
      - drv_gpio.c
      - drv_gpio.h
      - drv_log.h
      - drv_lptim.h
      - drv_pwm.c
      - drv_qspi.h
      - drv_sdio.h
      - drv_soft_i2c.h
      - drv_spi.h
      - drv_usart.c
      - drv_usart.h
      - drv_usbh.h
      - Kconfig
      - SConscript
    - STM32F4xx_HAL
    - Kconfig
  - packages
  - rt-thread [latest]
  - rtconfig.h
  - README.md

main.c | stm32f4xx_hal_msp.c | board.h | stm32f4xx_hal_conf.h | drv_pwm.c

```c
13 #ifdef RT_USING_PWM
14 #include "drv_config.h"
15 #include <drivers/rt_drv_pwm.h>
16
17 //#define DRV_DEBUG
18 #define LOG_TAG             "drv.pwm"
19 #include <drv_log.h>
20
21 #define MAX_PERIOD 65535
22 #define MIN_PERIOD 3
23 #define MIN_PULSE 2
24
25 extern void HAL_TIM_MspPostInit(TIM_HandleTypeDef *htim);
26
27 enum
28 {
29 #ifdef BSP_USING_PWM1
30     PWM1_INDEX,
31 #endif
32 #ifdef BSP_USING_PWM2
33     PWM2_INDEX,
34 #endif
35 #ifdef BSP_USING_PWM3
36     PWM3_INDEX,
37 #endif
38 #ifdef BSP_USING_PWM4
39     PWM4_INDEX,
40 #endif
41 #ifdef BSP_USING_PWM5
42     PWM5_INDEX,
43 #endif
44 #ifdef BSP_USING_PWM6
45     PWM6_INDEX,
46 #endif
47 #ifdef BSP_USING_PWM7
```

Outline | Build Targets

- C:\RT-ThreadStudio\workspace2\Servo_Contro
- board.h
- drv_config.h
- drivers/rt_drv_pwm.h
- LOG_TAG
- drv_log.h
- MAX_PERIOD
- MIN_PERIOD
- MIN_PULSE
- HAL_TIM_MspPostInit(TIM_HandleTypeDef*) : v
- (anonymous)
- stm32_pwm
- stm32_pwm_obj : struct stm32_pwm[]
- drv_pwm_control(struct rt_device_pwm*, int, v
- drv_ops : struct rt_pwm_ops
- drv_pwm_enable(TIM_HandleTypeDef*, struct r
- drv_pwm_get(TIM_HandleTypeDef*, struct rt_pv
- drv_pwm_set(TIM_HandleTypeDef*, struct rt_pv
- drv_pwm_control(struct rt_device_pwm*, int, v
- stm32_hw_pwm_init(struct stm32_pwm*) : rt_e
- pwm_get_channel(void) : void
- stm32_pwm_init(void) : int
- __rt_init_stm32_pwm_init : const init_fn_t

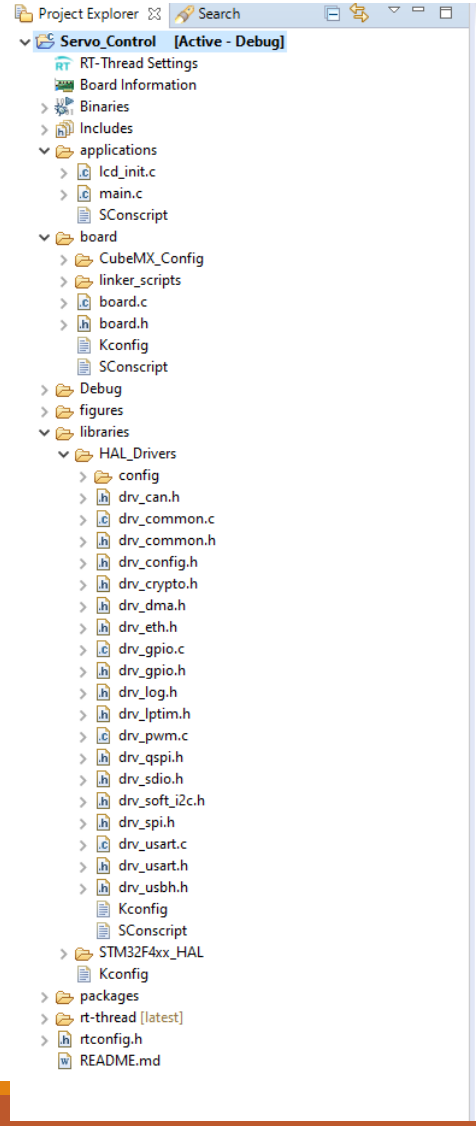
Problems | Tasks | Console | Properties

CDT Build Console [Servo_Control]

```
arm-none-eabi-gcc "../libraries/HAL_Drivers/drv_common.c"
arm-none-eabi-gcc "../libraries/HAL_Drivers/drv_gpio.c"
arm-none-eabi-gcc "../libraries/HAL_Drivers/drv_pwm.c"
arm-none-eabi-gcc "../libraries/HAL_Drivers/drv_usart.c"
arm-none-eabi-gcc "../board/CubeMX_Config/Src/stm32f4xx_hal_msp.c"
arm-none-eabi-gcc "../board/board.c"
arm-none-eabi-gcc "../applications/lcd_init.c"
arm-none-eabi-gcc "../applications/main.c"
linking...
arm-none-eabi-objcopy -O binary "rtthread.elf"  "rtthread.bin"
arm-none-eabi-size --format=berkeley "rtthread.elf"
   text    data     bss     dec     hex filename
 150080    2448    4512  157040   26570 rtthread.elf

             Used Size(B)          Used Size(KB)
Flash:        152528 B             148.95 KB
RAM:            6960 B               6.80 KB

12:50:34 Build Finished. 0 errors, 0 warnings. (took 11s.188ms)
```

Writable | Smart Insert | 44 : 22

12. We enter the MSP file of the microcontroller, where the PWM output pins are configured. By default we have the programming configured for timer 4 with another output pin.



stm32f4xx_hal_msp.c | main.c | board.h | stm32f4xx_hal_conf.h

```c
531 */
532 void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim_base)
533 {
534   GPIO_InitTypeDef GPIO_InitStruct = {0};
535   if(htim_base->Instance==TIM4)
536   {
537   /* USER CODE BEGIN TIM4_MspInit 0 */
538
539   /* USER CODE END TIM4_MspInit 0 */
540     /* Peripheral clock enable */
541     __HAL_RCC_TIM4_CLK_ENABLE();
542
543     __HAL_RCC_GPIOD_CLK_ENABLE();
544     /**TIM4 GPIO Configuration
545     PD12     ------> TIM4_CH1
546     */
547     GPIO_InitStruct.Pin = GPIO_PIN_12;
548     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
549     GPIO_InitStruct.Pull = GPIO_NOPULL;
550     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
551     GPIO_InitStruct.Alternate = GPIO_AF2_TIM4;
552     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
553
554   /* USER CODE BEGIN TIM4_MspInit 1 */
555
556   /* USER CODE END TIM4_MspInit 1 */
557   }
558
559 }
560
561 void HAL_TIM_MspPostInit(TIM_HandleTypeDef* htim)
562 {
563   GPIO_InitTypeDef GPIO_InitStruct = {0};
564   if(htim->Instance==TIM4)
565   {
566   /* USER CODE BEGIN TIM4_MspPostInit 0 */
567
568   /* USER CODE END TIM4_MspPostInit 0 */
569
570     __HAL_RCC_GPIOD_CLK_ENABLE();
571     /**TIM4 GPIO Configuration
572     PD13     ------> TIM4_CH2
573     */
574     GPIO_InitStruct.Pin = GPIO_PIN_13;
575     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
576     GPIO_InitStruct.Pull = GPIO_NOPULL;
577     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
578     GPIO_InitStruct.Alternate = GPIO_AF2_TIM4;
579     HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
580
```

```c
592 */
593 void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef* htim_base)
594 {
595   if(htim_base->Instance==TIM4)
596   {
597   /* USER CODE BEGIN TIM4_MspDeInit 0 */
598
599   /* USER CODE END TIM4_MspDeInit 0 */
600     /* Peripheral clock disable */
601     __HAL_RCC_TIM4_CLK_DISABLE();
602
603     /**TIM4 GPIO Configuration
604     PD12     ------> TIM4_CH1
605     PD13     ------> TIM4_CH2
606     */
607     HAL_GPIO_DeInit(GPIOD, GPIO_PIN_12|GPIO_PIN_13);
608
609   /* USER CODE BEGIN TIM4_MspDeInit 1 */
610
611   /* USER CODE END TIM4_MspDeInit 1 */
612   }
613
614 }
```

13. Therefore, the configuration of timers 2 and 3 is carried out, activating and deactivating the clock configuration in their respective functions.
Also in the initialization configuration of the pins. The function identifies the type of timer that is being used, for which the corresponding pins are initialized in their respective channels. These pins are PA15, PA2, PB4 and PB5.

```c
void HAL_TIM_Base_MspInit(TIM_HandleTypeDef* htim_base)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(htim_base->Instance==TIM2)
    {

          /* Peripheral clock enable */
        __HAL_RCC_TIM2_CLK_ENABLE();

      }
   else if(htim_base->Instance==TIM3)
     {

          /* Peripheral clock enable */
          __HAL_RCC_TIM3_CLK_ENABLE();

     }

}
```

```c
void HAL_TIM_Base_MspDeInit(TIM_HandleTypeDef* htim_base)
{
    if(htim_base->Instance==TIM2)
    {
            /* Peripheral clock disable */
     __HAL_RCC_TIM2_CLK_DISABLE();

     }
     else if(htim_base->Instance==TIM3)
     {
            /* Peripheral clock disable */
     __HAL_RCC_TIM3_CLK_DISABLE();

     }
}
```

```c
void HAL_TIM_MspPostInit(TIM_HandleTypeDef* htim)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    if(htim->Instance==TIM2)
     {

      __HAL_RCC_GPIOA_CLK_ENABLE();
      /**TIM2 GPIO Configuration
      PA15     ------> TIM2_CH1
      PA2      ------> TIM2_CH3

      */
      GPIO_InitStruct.Pin = GPIO_PIN_15|GPIO_PIN_2;
      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
      GPIO_InitStruct.Pull = GPIO_NOPULL;
      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
      GPIO_InitStruct.Alternate = GPIO_AF1_TIM2;
      HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

       }
      else if(htim->Instance==TIM3)
      {


      __HAL_RCC_GPIOB_CLK_ENABLE();
      /**TIM3 GPIO Configuration
      PB4      ------> TIM3_CH1
      PB5      ------> TIM3_CH2

           */
      GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_5;
      GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
      GPIO_InitStruct.Pull = GPIO_NOPULL;
      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
      GPIO_InitStruct.Alternate = GPIO_AF2_TIM3;
      HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);


      }

}
```

14. In the main file, the corresponding Macros are defined, the PWM that will be used and the respective channels. PWM device structures are created to perform initialization.

workspace2 - Servo_Control/applications/main.c - RT-Thread Studio

File Edit Source Navigate Project Run Window Help

Project Explorer | Search

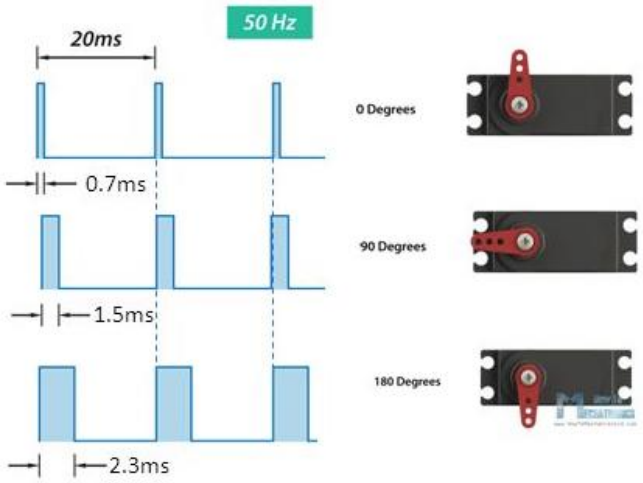
- Servo_Control [Active - Debug]
  - RT-Thread Settings
  - Board Information
  - Binaries
  - Includes
  - applications
    - lcd_init.c
    - main.c
    - SConscript
  - board
    - CubeMX_Config
    - linker_scripts
    - board.c
    - board.h
    - Kconfig
    - SConscript
  - Debug
  - figures
  - libraries
    - HAL_Drivers
    - STM32F4xx_HAL
    - Kconfig
  - packages
  - rt-thread [latest]
  - rtconfig.h
  - README.md

*main.c | stm32f4xx_hal_msp.c | board.h | stm32f4xx_hal_conf.h | drv_pwm.c

```c
/*
 * Copyright (c) 2006-2018, RT-Thread Development Team
 *
 * SPDX-License-Identifier: Apache-2.0
 *
 * Change Logs:
 * Date           Author       Notes
 * 2018-11-06     SummerGift   first version
 * 2019-04-09     WillianChan  add stm32f469-st-disco bsp
 */

#include <rtthread.h>
#include <rtdevice.h>
#include <board.h>

/* defined the LED1 pin: PG6 */
#define PWM_DEV_NAME_2        "pwm2" /* PWM 2  */
#define PWM_DEV_CHANNEL_2_1     1 /* Channel 2  PWM2   */
#define PWM_DEV_CHANNEL_2_3     3 /* Channel 3  PWM2  */

#define PWM_DEV_NAME_3       "pwm3" /* PWM 3 */
#define PWM_DEV_CHANNEL_3_1     1 /* Channel 1  PWM3   */
#define PWM_DEV_CHANNEL_3_2     2 /* Channel 2  PWM3   */


struct rt_device_pwm *pwm_dev2; /* Handle PWM 2  */

struct rt_device_pwm *pwm_dev3; /* Handle PWM 3  */

void main(void)
{

}
```

15. In the main function, uint32 variables corresponding to the period of the PWM signal in units of ns are defined, and the servo positions are based on the servo motor specification.
In this case the servomotor works at a frequency of 50Hz. Therefore, the value of the period is as the frequency is obtained, in the same way the value of the positions are according to the necessary pulse width.

```
void main(void)
{
    rt_uint32_t period, pos1,pos2;

    period = 20000000; /* Period for servo MG90S */

    pos1 = 700000; /* PWM Pulse   blunt   wide   degree   value ,  single   position   by   nanosecond ns */

    pos2 = 2000000;/* PWM Pulse   blunt   wide   degree   value ,  single   position   by   nanosecond ns */

}
```



$$frecuency = \frac{1}{period} = \frac{1}{20000000ns} = 50\text{Hz}$$

16. The search function is used to find device identifiers according to the name of the PWM device, which were defined in the macros
For each of the PWM channels, its values are set, the device to be managed, the respective channel, the period and the pulse.
In addition, the activation of the channels of the devices is carried out.

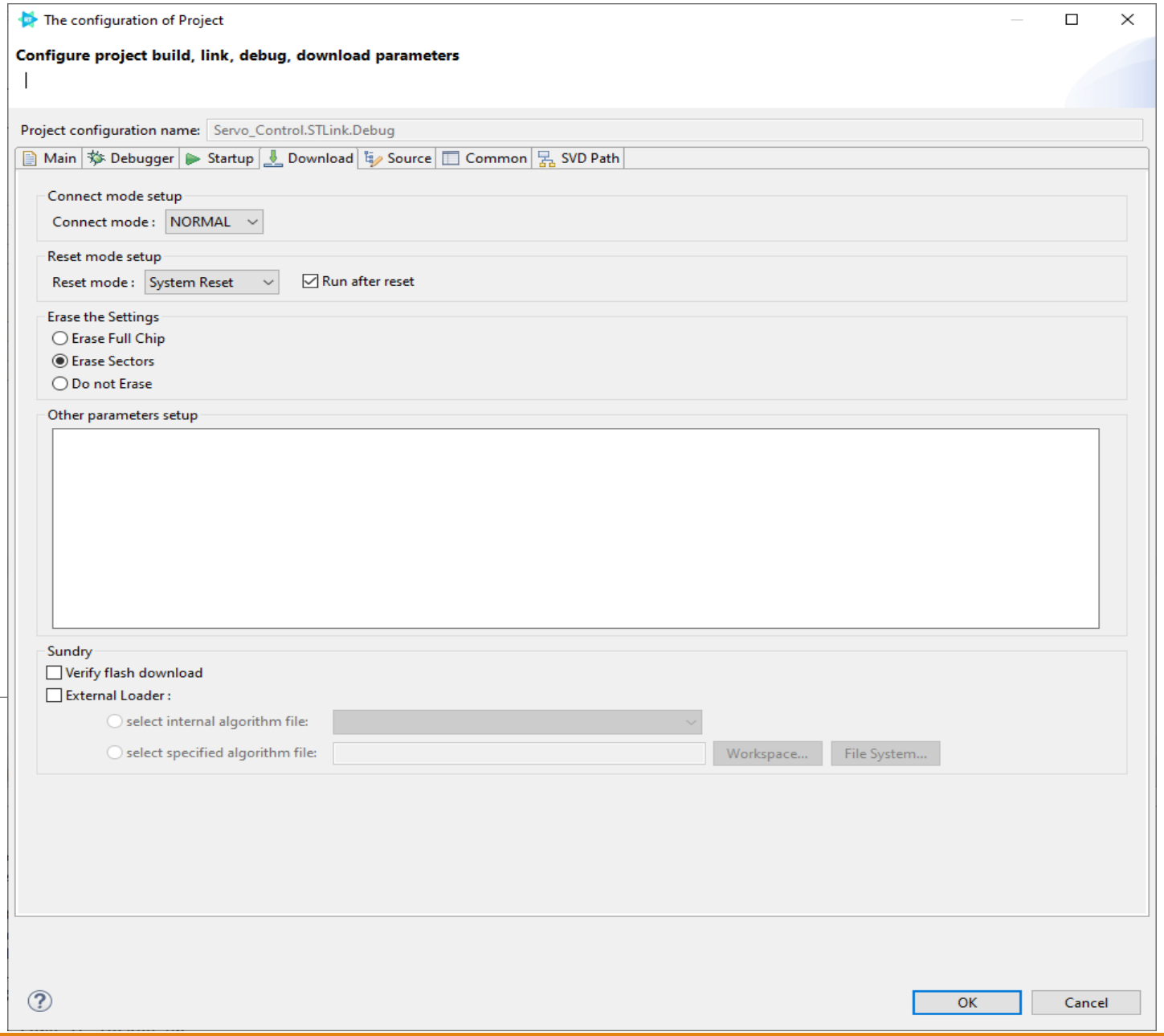
```c
void main(void)
{
    rt_uint32_t period, pos1,pos2;

    period = 20000000; /* Period for servo MG90S */

    pos1 = 700000; /* PWM Pulse   blunt   wide   degree   value ,  single   position   by   nanosecond ns */

    pos2 = 2000000;/* PWM Pulse   blunt   wide   degree   value ,  single   position   by   nanosecond ns */

    pwm_dev2 = (struct rt_device_pwm *)rt_device_find(PWM_DEV_NAME_2);

    pwm_dev3 = (struct rt_device_pwm *)rt_device_find(PWM_DEV_NAME_3);

    /*  Set up PWM  */

    rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_1, period, pos2);
    rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_3, period, pos1);
    rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_1, period, pos2);
    rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_2, period, pos1);

    /*  Enable Channel  */
    rt_pwm_enable(pwm_dev2, PWM_DEV_CHANNEL_2_1);
    rt_pwm_enable(pwm_dev2, PWM_DEV_CHANNEL_2_3);
    rt_pwm_enable(pwm_dev3, PWM_DEV_CHANNEL_3_1);
    rt_pwm_enable(pwm_dev3, PWM_DEV_CHANNEL_3_2);
}
```

17. An infinite loop is established, in which the position of the fingers is changed in order to open and close the hands. This change is made every 5 seconds, and the status of the fingers is printed on the terminal. Some of the outputs are inverted, since the servomotors have different positions.

```
while (1)
    {

        rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_1, period, pos2);/*Thumb*/
        rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_3, period, pos1);/*Middle*/
        rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_1, period, pos2);/* Index*/
        rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_2, period, pos1);/*Ring / Little */


        rt_kprintf("Open Fingers \n");

        rt_thread_mdelay(5000);

        rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_1, period, pos1);/*Thumb*/
        rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_3, period, pos2);/*Middle*/
        rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_1, period, pos1);/* Index*/
        rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_2, period, pos2); /*Ring / Little */



        rt_kprintf("Close Fingers \n");
        rt_thread_mdelay(5000);


    }
```

Before loading the program to the board, it is established that a System Reset is performed.

18. In the download configuration of the program, it is specified that a system reset is performed before uploading it to the board.



19. The program is loaded onto the board and we can see on the console that the program is running. If the terminal is opened, the outputs are displayed indicating the position of the fingers.

workspace2 - Servo_Control/applications/main.c - RT-Thread Studio

File Edit Source Navigate Project Run Window Help

Project Explorer

- Servo_Control
  - RT-Thread Settings
  - Board Information
  - Binaries
  - Includes
  - applications
    - lcd_init.c
    - main.c
    - SConscript
  - board
    - CubeMX_Config
    - linker_scripts
    - board.c
    - board.h
    - Kconfig
    - SConscript
  - Debug
  - figures
  - libraries
    - HAL_Drivers
    - STM32F4xx_HAL
    - Kconfig
  - packages
  - rt-thread [latest]
  - rtconfig.h
  - README.md

main.c | stm32f4xx_hal_msp.c | board.h | stm32f4xx_hal_conf.h | drv_pwm.c

```c
pos2 = 2000000;/* PWM Pulse blunt wide degree value , single position by nanosecond ns */

pwm_dev2 = (struct rt_device_pwm *)rt_device_find(PWM_DEV_NAME_2);

pwm_dev3 = (struct rt_device_pwm *)rt_device_find(PWM_DEV_NAME_3);

/* Set up PWM */

rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_1, period, pos2);
rt_pwm_set(pwm_dev2, PWM_DEV_CHANNEL_2_3, period, pos1);
rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_1, period, pos2);
rt_pwm_set(pwm_dev3, PWM_DEV_CHANNEL_3_2, period, pos1);

/* Enable Channel */
rt_pwm_enable(pwm_dev2, PWM_DEV_CHANNEL_2_1);
rt_pwm_enable(pwm_dev2, PWM_DEV_CHANNEL_2_3);
rt_pwm_enable(pwm_dev3, PWM_DEV_CHANNEL_3_1);
rt_pwm_enable(pwm_dev3, PWM_DEV_CHANNEL_3_2);
```

Outline | Build Targets

- C:\RT-ThreadStudio\workspace2\Servo_Control\rtconfig_preinc.h
- rtthread.h
- rtdevice.h
- board.h
- PWM_DEV_NAME_2
- PWM_DEV_CHANNEL_2_1
- PWM_DEV_CHANNEL_2_3
- PWM_DEV_NAME_3
- PWM_DEV_CHANNEL_3_1
- PWM_DEV_CHANNEL_3_2
- pwm_dev2 : struct rt_device_pwm*
- pwm_dev3 : struct rt_device_pwm*
- main(void) : void

Problems | Tasks | Console | Properties

Log Console

```
ST-LINK SN  : 066CFF505567494867245644
ST-LINK FW  : V2J40M27
Board       : 32F469IDISCOVERY
Voltage     : 3.24V
SWD freq    : 4000 KHz
Connect mode: Normal
Reset mode  : Software reset
Device ID   : 0x434
Revision ID : Rev A
Device name : STM32F469xx/F467xx
Flash size  : 2 MBytes
Device type : MCU
Device CPU  : Cortex-M4
Memory Programming ...
Opening and parsing file: rtthread.elf
  File          : rtthread.elf
  Size          : 152868 Bytes
  Address       : 0x08000000
Erasing memory corresponding to segment 0:
Erasing internal memory sectors [0 5]
Download in Progress:
Progress: 100%
File download complete
Time elapsed during download operation: 00:00:03.937
MCU Reset
Software reset is performed
RUNNING Program ...
  Address:      : 0x8000000
Application is running
Start operation achieved successfully
completed, elapse time :4091ms.
```

Problems | Tasks | Console | Properties | Terminal

Daniel

```
 \ | /
- RT -     Thread Operating System
 / | \     5.0.0 build Oct 20 2022 13:51:50
 2006 - 2022 Copyright by RT-Thread team
Open Fingers
msh />Close Fingers
Open Fingers
Close Fingers
Open Fingers
```